Минобрнауки России
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Волгоградский государственный технический университет»

Факультет Электроники и вычислительной техники

Кафедра Электронно-вычислительные машины и системы

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе (проекту)

по дисциплине Системы обработки больших данных

на тему: Исследование датасета рейтинга недвижимости в Англии с использованием фреймворка Apache Spark

Студент Ефременков Илья Евгеньевич
(фамилия, имя, отчество)

Группа САПР-1.3

Руководитель работы (проекта) _____ П.Д. Кравченя
(подпись и дата подписания) (инициалы и фамилия)

Члены комиссии:

_____ _____
(подпись и дата подписания) (инициалы и фамилия)

_____ _____
(подпись и дата подписания) (инициалы и фамилия)

_____ _____
(подпись и дата подписания) (инициалы и фамилия)

Нормоконтроллер _____ П.Д. Кравченя
(подпись и дата подписания) (инициалы и фамилия)

Волгоград 2025

Минобрнауки России
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Волгоградский государственный технический университет»

Факультет Электроники и вычислительной техники

Направление (специальность) Информатика и вычислительная техника

Кафедра Электронно-вычислительные машины и системы

Дисциплина Системы обработки больших данных

Утверждаю
Зав. кафедрой А.Е. Андреев

«____» _____ 20 ____ г.

ЗАДАНИЕ
на курсовую работу (проект)

Студент Ефременков Илья Евгеньевич
(фамилия, имя, отчество)

Группа САПР-1.3

1. Тема: Исследование датасета рейтинга недвижимости в Англии с использованием фреймворка Apache Spark

Утверждена приказом от «___» _____ 20 ___ г., № _____.

2. Срок представления работы (проекта) к защите «___» _____ 20 ___ г.

3. Содержание расчётно-пояснительной записки: РАЗВЕДОЧНЫЙ АНАЛИЗ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ PYSPARK; МАШИННОЕ ОБУЧЕНИЕ НА БОЛЬШИХ ДАННЫХ.

4. Перечень графического материала: _____

5. Дата выдачи задания «___» _____ 20 ___ г.

Руководитель работы (проекта) _____ П.Д. Кравченя
(подпись и дата подписания) (инициалы и фамилия)

Задание принял к исполнению _____ И.Е. Ефременков
(подпись и дата подписания) (инициалы и фамилия)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Во введении сначала дается краткая характеристика области, в которой выполнена работа (1 – 3 предложения). Затем обосновывается актуальность работы.

Данная работа выполнена в среде Appache Spark - - это мощный open-source фреймворк для обработки больших данных, который применяется в различных областях: В целом, область применения Apache Spark очень широка и зависит от потребностей конкретной организации или проекта.

Далее идут фразы, которые лучше повторить дословно:

В связи с этим целью данной работы являлось ... (цель должна быть одна). ?????????????(в двух лабах их несколько)

Для достижения поставленной цели решались следующие задачи:

1. Познакомиться с понятием «большие данные» и способами их обработки.
2. Познакомиться с инструментом Apache Spark и возможностями, которые он предоставляет для обработки больших данных.
3. Получить представление об инструментах экосистемы Hadoop: HDFS и YARN.
4. Поработать с табличным форматом для больших данных Apache Iceberg.
5. Получить навыки выполнения разведочного анализа данных использованием pyspark.

В конце введения следует добавить описание структуры курсовой работы. Например:

В первом разделе рассмотрена более подробно постановка задачи разведовочного анализа датасета с использование фреймворка Apache Spark и библиотеки SparkML

Во втором разделе расмотрены более подробно задачи линейной регрессии и бинарной классификации

... В третьем разделе ??????????????

... В заключении работы сформулированы общие выводы ???????

1 РАЗВЕДОЧНЫЙ АНАЛИЗ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ PYSPARK

1.1 Постановка задачи разведочного анализа

Разведочный анализ датасета - это первый этап в процессе анализа данных, который состоит из нескольких шагов. Цель этого этапа - получить общее представление о данных и их характеристиках, чтобы понять, какие методы анализа могут быть применены.

Основные цели разведочного анализа датасета включают:

1. Определение структуры данных: Разведочный анализ помогает определить, как выглядит ваш датасет – сколько столбцов (признаков) и строк (данных), какие типы данных используются (числовые, категориальные и т.д.).
2. Понимание распределения данных: Этот этап включает в себя проверку на наличие отсутствующих значений, выбросов или аномалий в данных.
3. Изучение основных характеристик данных: Определение среднего значения, моды, медианы и стандартного отклонения для числовых признаков; подсчет количества уникальных категорий для категориальных признаков.
4. Выявление взаимосвязей между переменными: Этот этап помогает понять, какие признаки могут влиять на другие в вашем датасете.
5. Визуализация данных: Построение графиков и диаграмм для наглядного представления данных и их распределения.

Все эти шаги важны для того, чтобы понять ваши данные и выбрать правильные методы для дальнейшего анализа или машинного обучения.

Таблица 1 – Таблица признаков

Признак	Расшифровка признака
LMK_KEY	Первичный ключ
ADDRESS	Адрес
CURRENT_ENERGY_EFFICIENCY	Эффективность энергии
PROPERTY_TYPE	Тип квартиры
INSPECTION_DATE	Дата инспекции
HEATING_COST_CURRENT	Затраты на обогрев
HOT_WATER_COST_CURRENT	Затраты на горячую воду
TOTAL_FLOOR_AREA	Площадь
NUMBER_HABITABLE_ROOMS	Количество обитаемых комнат
NUMBER_HEATED_ROOMS	Количество комнат с подогревом

1.2 Описание датасета

В приведенном датасете рассматриваются квартиры и их атрибуты для расчёта платы, такие как: площадь квартиры, её тип, использованное количество энергии, расход горячей воды и т.д. Таким образом из всех имеющихся признаков определяется рейтинг квартиры. Сам датасет весит более 20 Гб, но для работы с ним нам пришлось его обрезать до 5 Гб [**https://www.kaggle.com/datasets/tyagia1/epcratingsenglandjuly203**]. Выбраны определенные 10 столбцов признаков для выполнения разведочного анализа.

1.3 Определение пропущенных значений

При анализе мы предполагали, где могут встречаться пропущеннные значения. С помощью команды count_nulls мы определяли количества пропущенных значений в том или ином столбце. Ниже приведен список признаков, где пропущенные значения были недопустимы

А здесь – аналогичный нумерованный список:

```
Минимальное значение:          -46.00
Среднее значение:              656.27
Среднеквадратичное отклонение: 511.08
Первый квартиль:               369.00
Медиана:                       552.00
Третий квартиль:               796.00
Максимальное значение:         58374.00
```

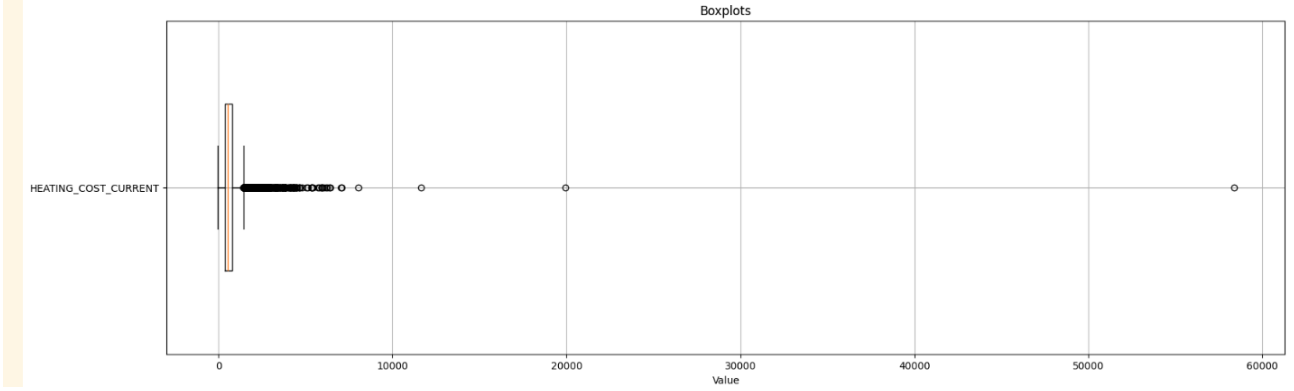


Рисунок 1 – Выбросы в HEATING_COST_CURRENT

1. ADDRESS3;
2. HEATING_COST_CURRENT;
3. NUMBER_HABITABLE_ROOMS
4. NUMBER_HEATED_ROOMS

После определения количества пропущенных значений мы работали с этими критериями мы их обрабатывали разными способами.

У ADDRESS3 более 90% значений пропущены, этот столбец был удален в целях упрощения работы с датасетом.

```
df = df.drop("ADDRESS3")
df.show()
```

В аналогичном случае можно было бы заменить пропущенные значения на значение Unknown

```
df = df.fillna({"PROPERTY\_TYPE": "Unknown"})
count\_nulls(data=df, column\_name="PROPERTY\_TYPE")
```

В столбце HEATING_COST_CURRENT было пропущенно значений меньше половины, значит, можно попытаться обработать эти данные. Была создана функция позволяющая рассчитывать статистические показатели данных в столбцах и строить диаграмму ”ящик с усами” для оценки наличия выбросов: ”plot_boxplots”. В результате получился boxplot с сильными выбросами в

нескольких точках. Удалим строки, их содержащие, и убедимся, что потеряна небольшая часть данных и заменим пропуски средним значением признака.

```
df.filter(col("HEATING\_COST\_CURRENT") > 8000).count()
df = df.filter(col("HEATING\_COST\_CURRENT") < 8000)
mean\_cost = df.select(mean(col("HEATING\_COST\_CURRENT"))).collec
mean\_cost
df = df.fillna({"HEATING\_COST\_CURRENT": mean\_cost})
```

В столбцах NUMBER_HABITABLE_ROOMS и NUMBER_HEATED_ROOMS содержатся пропущенные значения с типом float, заменим их на значения 0.0

```
df = df.fillna({"NUMBER\_HABITABLE\_ROOMS": 0.0})
df = df.fillna({"NUMBER\_HEATED\_ROOMS": 0.0})
```

1.4 Расчет корреляции между количественными признаками

Рассчет корреляции между количественными признаками — это процесс, который позволяет определить степень и направление взаимосвязи между двумя или более переменными. Корреляция отражает насколько часто значения одной переменной изменяются вместе с изменениями в другой переменной.

Существует два основных типа корреляции:

1. Позитивная корреляция: Это ситуация, когда значения двух переменных меняются в одно и то же направление. Например, если при увеличении одной переменной (например, времени) увеличивается и другая (например, расстояние), это будет показывать положительную корреляцию.
2. Негативная корреляция: Это ситуация, когда значения двух переменных меняются в противоположных направлениях. Например, если при увеличении одной переменной (например, времени) уменьшается другая (например, расстояние), это будет показывать негативную корреляцию.

Рассчитать корреляцию между количественными признаками можно с помощью различных статистических методов. Наиболее распространенным из них является коэффициент корреляции Пирсона, который показывает степень и направление линейной связи между двумя переменными.

Коэффициент корреляции Пирсона (r) вычисляется по формуле:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

В большинстве случаев используется коэффициент корреляции Пирсона для измерения линейной связи между двумя количественными признаками. Коэффициент Пирсона, обозначенный как r, варьируется от -1 до 1:

1. Если r близок к 1, то связь между переменными очень сильная и положительная.
2. Если r близок к -1, то связь между переменными очень сильная и негативная.
3. Если r близок к 0, то между переменными нет линейной связи.

Важно отметить, что корреляция не говорит о причине и последствии между двумя переменными; она просто показывает, насколько сильно значения одной переменной изменяются в зависимости от значений другой.

Корреляционные коэффициенты часто используются для анализа и визуализации данных, чтобы определить взаимосвязь между различными признаками. Они могут помочь в выборе переменных для моделей машинного обучения или в выявлении закономерностей в данных.

Код расчёта:

```
def compute_and_visualize_correlation_matrix(data: DataFrame,
                                             columns: list[str]) -
    """
    Вычисляет и визуализирует корреляционную матрицу для указанных
    колонок в DataFrame PySpark.

    Args:
```

```
        df (DataFrame): DataFrame PySpark.
        columns (list[str]): Список колонок для вычисления корреля

    Returns:
        None
    """
    # Вычисление корреляционной матрицы
    corr_matrix = {}
    for col1 in columns:
        corr_matrix[col1] = {}
        for col2 in columns:
            corr_value = data.select(corr(col1, col2)).collect()[0
            corr_matrix[col1][col2] = corr_value

    # Преобразование корреляционной матрицы в DataFrame Pandas для
    corr_matrix_pd = pd.DataFrame(corr_matrix)

    # Построение и визуализация корреляционной матрицы
    plt.figure(figsize=(10, 8))
    sns.heatmap(corr_matrix_pd, annot=True, cmap='coolwarm', linew
    plt.title('Correlation Matrix')
    plt.show()
```

1.5 Выводы

В результате полученной работы была разработана корреляционная матрица, демонстрирующая наличие корреляции между количественными признаками.

Здесь наблюдается наибольшая зависимость обитаемых и обогреваемых комнат от площади самой квартиры.

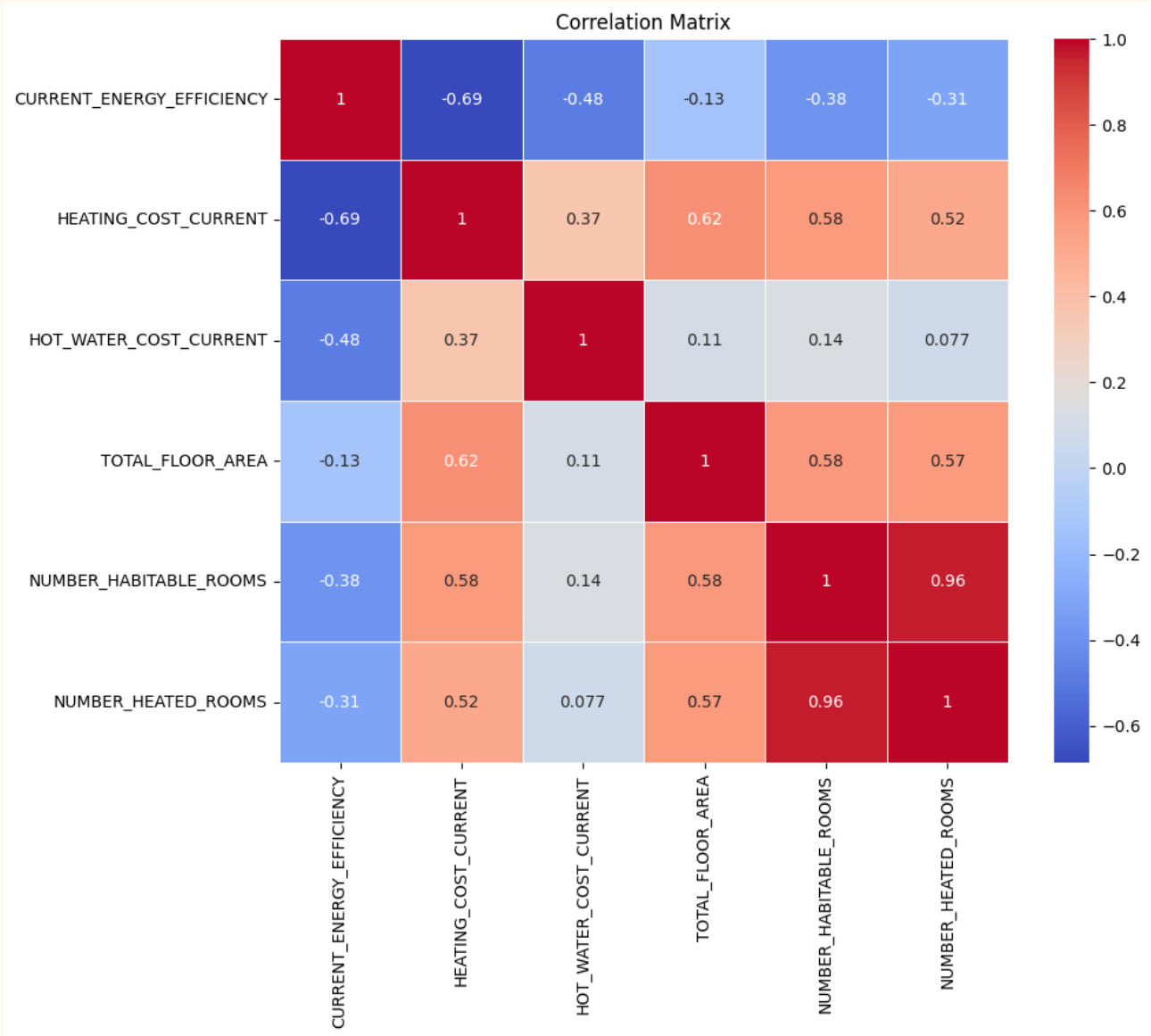


Рисунок 2 – Корреляционная матрица датасета

2 МАШИННОЕ ОБУЧЕНИЕ НА БОЛЬШИХ ДАННЫХ

В этой главе подробно рассматриваются задачи линейной регрессии и бинарной классификации больших данных.

2.1 Задача регресии

2.1.1 Постановка задачи регрессии

Постановка задачи линейной регрессии заключается в построении математической модели, которая позволяет предсказывать значение зависимой переменной на основе независимых переменных. Целью является минимизация ошибок и нахождение коэффициентов, которые обеспечивают наиболее точные прогнозы. В данном датасете мы, исходя из разведочного анализа, выяснили предсказываем значение зависимого столбца TOTAL_FLOOR_AREA. Также нужно рассчитать оценку качества модели.

Для датасета, заданного представленными колонками, требуется построить модель линейной регрессии для оценки **Площадь квартиры** по всем колличественным и категориальным признакам.

Для оценки качества обучения следует использовать метрики $RMSE$ и R^2.

2.1.2 Решение задачи регрессии

Подготовка и кодирование признаков.

Для корректной работы трансформеров преобразуем столбцы ‘HEATING_COST_CURRENT‘, ‘HOT_WATER_COST_CURRENT‘, ‘NUMBER_HA
‘NUMBER_HEATED_ROOMS‘ к типу ‘DoubleType‘.

```
df = df.withColumn("HEATING_COST_CURRENT", col("HEATING_COST_CURRE
df = df.withColumn("HOT_WATER_COST_CURRENT", col("HOT_WATER_COST_C
```

```
df = df.withColumn("NUMBER_HABITABLE_ROOMS", col("NUMBER_HABITABLE
df = df.withColumn("NUMBER_HEATED_ROOMS", col("NUMBER_HEATED_ROOMS
```

Отделим от датасета некоторую часть объёмом примерно 1000 строк, и сохраним её на диске как локальный ‘csv‘-файл. Он понадобится в следующей лабораторной работе.

```
def save_sample_to_csv(data: DataFrame, file_path: str,
                       sample_size: int = 1000) -> DataFrame:
```

Определяем путь для сохранения ‘csv‘-файла.

```
path = "/home/user6/Efremenkov_directory/dataset/data/epc_cut_reg.
df = save_sample_to_csv(data=df, file_path=path, sample_size=1000)
```

Оцениваем, сколько строк в датасете осталось.

```
df.count()
```

Разделим датасет на обучающую и тестовую выборки.

```
train_df, test_df = df.randomSplit([0.8, 0.2])
```

Понятно, что **ключ и адрес** квартиры не оказывает влияния на тип недвижимости. Использовать его в модели нет смысла. Остальные признаки сгруппируем по их типу:

* **Категориальные** признаки не содержат большого количества категорий, закодируем их ‘one-hot‘-кодировкой. * **Бинарные** признаки представлены значениями ‘true‘ / ‘false‘, которые могут быть интерпретированы как единица и нуль. Поэтому, в кодировании не нуждаются. * **Количественные** признаки нужно нормализовать / стандартизировать, перед тем, как передавать их в модель.

```
categorical_features = [ "PROPERTY_TYPE" ]
numeric_features = [
    "CURRENT_ENERGY_EFFICIENCY", "HEATING_COST_CURRENT", "HOT_WATE
]
```

Создадим конвейер обработки данных, включающий модель линейной регрессии.

```
def create_pipeline(categorical_features: list[str],
                    numeric_features: list[str],
                    #binary_features: list[str], binarized_col: st
                    #threshold: float,
                    label_col: str, max_iter: int) -> Pipeline: ..

pipeline = create_pipeline(categorical_features=categorical_featur
                           numeric_features=numeric_features,
                           #binary_features=binary_features,
                           label_col="TOTAL_FLOOR_AREA",
                           max_iter=15)
```

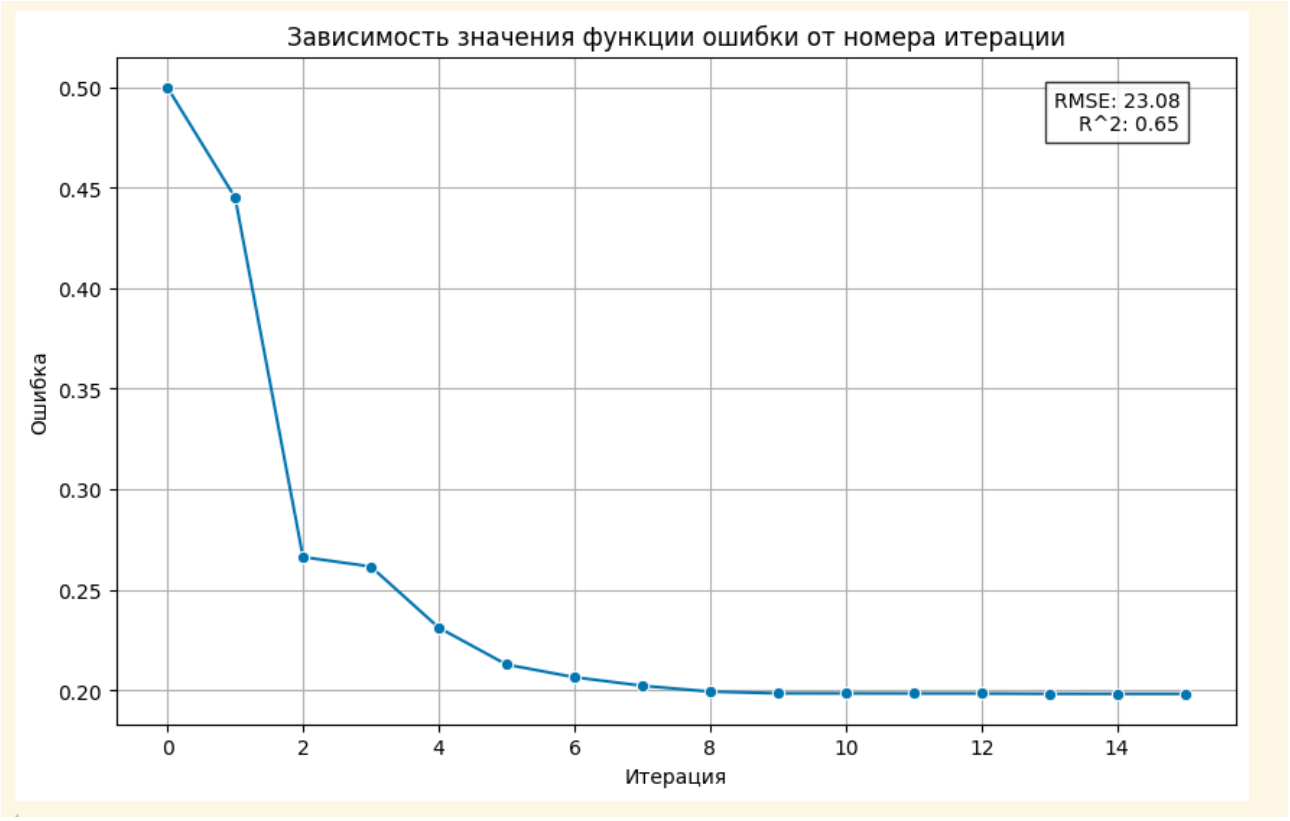
Обучение модели. Выполним **подбор гиперпараметров** модели линейной регрессии с помощью кросс-валидации на сетке. Создаем сетку параметров для кросс-валидации, получив объект ‘LinearRegression‘ из конвейера. Создаем экземпляр ‘RegressionEvaluator‘ для оценки модели. Создаем объект ‘CrossValidator‘.

```
param_grid = ParamGridBuilder() \
    .addGrid(pipeline.getStages()[-1].regParam, [0.01, 0.1, 1.0])
    .addGrid(pipeline.getStages()[-1].elasticNetParam, [0.0, 0.5,
    .build()

cv_evaluator = RegressionEvaluator(labelCol="TOTAL_FLOOR_AREA",
                                   predictionCol="prediction",
                                   metricName="rmse")

cross_validator = CrossValidator(estimator=pipeline,
                                 estimatorParamMaps=param_grid,
                                 evaluator=cv_evaluator,
                                 numFolds=5)
```

Обучаем модель конвейера с использованием кросс-валидации.




```
cv_model = cross_validator.fit(train_df)
```

Выведем параметры **лучшей** модели, определенной в ходе кросс-валидации.

```
def get_best_model_params(cv_model: CrossValidatorModel) -> dict[s
for key, value in get_best_model_params(cv_model=cv_model).items()
    print(f"{key}: {value}")
```

2.1.3 Анализ полученных результатов

Визуализируем изменение ошибки модели в ходе обучения и рассчитаем метрики на обучающем датасете.

```
def plot_training_summary(cv_model: DataFrame) -> None: ... Прилож
plot_training_summary(cv_model)
```

Проверка обобщающей способности модели Выполним предсказания на тестовой выборке. Перегруппируем колонки датафрейма, переставив столбец

с площадью квартиры в конец, чтобы его значения было удобно сравнивать с предсказанными.

```
# Получаем датасет предсказаний
test_df_predictions = cv_model.transform(test_df)

# Извлекаем список колонок, устанавливаем цену на последнее место
right_columns_order = test_df_predictions.columns
right_columns_order.remove("TOTAL_FLOOR_AREA")
right_columns_order.append("TOTAL_FLOOR_AREA")

# Изменяем последовательность колонок и выводим датафрейм
test_df_predictions = test_df_predictions.select(*right_columns_or
test_df_predictions.show()
```

Создадим функцию оценки модели: расчета метрик для некоторого датасета, как правило, тестового.

```
def evaluate_model(data: DataFrame, metric_name: str) -> float:
```

Оценим модель на тестовой выборке.

```
test_rmse = evaluate_model(test_df_predictions, "rmse")
test_r2 = evaluate_model(test_df_predictions, "r2")

print(f"RMSE on test data: {test_rmse:.2f}")
print(f"R^2 on test data: {test_r2:.2f}")
```

Результат:

1. RMSE on test data: 22.95
2. $R2$ on test data: 0.65

Метрики весьма неплохие для данной модели.

2.2 Задача бинарной классификации

В данной части работы рассмотрены:

– подготовка признаков для рашения задачи **градиентного бустинга** на деревьях решений;
– создание и обучение модели градиентного бустинга;
– оценка качества модели.

2.2.1 Постановка задачи бинарной классификации

Постановка задачи бинарной классификации включает определение, какие данные будут использоваться для обучения модели, выбор метрик для оценки качества модели и определение целевой переменной.

1. Определение данных

   Для построения модели необходимо иметь набор данных, который содержит:

   а) Входные признаки (features): Эти данные используются для прогнозирования целевой переменной.

   б) Целевая переменная (target variable): Эта переменная представляет собой класс, к которому относится каждый объект в наборе данных. В случае бинарной классификации это обычно два значения, например, 0 и 1.

2. Выбор метрик для оценки качества модели

   Для оценки качества модели на этапе тестирования используются различные метрики. Основные метрики в бинарной классификации:

   а) Точность (Precision): Процент объектов, которые действительно принадлежат положительному классу и были корректно определены моделью. [$Precision = \frac{TP}{TP+FP}$] Где (TP) — истинные положительные, а (FP) — ложные положительные.

   б) Полнота (Recall): Процент объектов, которые действительно принадлежат положительному классу и были корректно определены моделью. [$Recall = \frac{TP}{TP+FN}$] Где (TP) — истинные положительные, а (FN) — ложные отрицательные.

в) F1-мера: Среднее гармоническое точности и полноты. Это метрика, которая учитывает обе ошибки (FP и FN). [$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$]

г) ROC-AUC (Area Under the Receiver Operating Characteristic Curve): Площадь под кривой ROC, которая показывает, как хорошо модель различает положительные и отрицательные классы при различных порогах вероятности.

1) Кривая ROC строится на основе значений вероятностей принадлежности положительному классу для всех объектов в наборе данных.

3. Определение целевой переменной

Целевая переменная в бинарной классификации должна быть категориальной и иметь всего два уникальных значения, например, ”0” и ”1”. Значения могут представлять собой различные классы или категории, такие как ”заболевал” vs ”не заболевал”, ”покупает” vs ”не покупает”.

Шаги построения модели

1. Подготовка данных:
   а) Разделить данные на наборы для обучения и тестирования.
   б) Нормализовать или стандартизировать признаки.
   в) Обработать пропущенные значения (например, удалить строки с пропущенными значениями).
2. Выбор модели:
   а) Выбрать подходящую модель бинарной классификации, например, логистическая регрессия, случайный лес или нейронную сеть.
3. Обучение модели:
   а) Обучить модель на наборе данных для обучения.
4. Оценка качества модели:
   а) Оценить качество модели на наборе данных для тестирования.

б) Использовать выбранные метрики (точность, полнота, F1-мера, ROC-AUC) для оценки производительности модели.

5. Кросс-валидация:
    а) Применить кросс-валидацию для улучшения надежности оценок качества модели.
6. Оптимизация параметров:
    а) Оптимизировать параметры модели с помощью методов, таких как градиентный спуск или случайный поиск.
7. Интерпретация результатов:
    а) Понять, какие признаки влияют на вероятность открытия нового счета.
    б) Делать выводы о том, насколько хорошо модель предсказывает целевую переменную.

Для датасета, заданного представленными колонками, требуется построить модель градиентного бустинга на деревьях решений для оценки факта того, является ли автомобиль сертифицированным, по всем остальным признакам.

Для оценки качества обучения следует использовать метрики ‘Precision‘ и ‘Recall‘. Оценить максимально возможное значение точности при полноте не менее 60

2.2.2 Решение задачи бинарной классификации

Аналогично задаче регрессии.

2.2.3 Анализ полученных результатов

Определим вероятность – границу разделения, при которой ‘Recall‘ не меньше 60%.

```
threshold_probability = pd_dataframe[pd_dataframe['TPR'] >= 0.60][
print(f"Вероятность -- граница разделения, при которой TPR не мены
```

Вероятность – граница разделения, при которой TPR не меньше 60%: 0.70

Рассчитаем метрики на тестовом датасете повторно, с учетом вычисленного ‘threshold‘ для вероятности.

```
cv_model.bestModel.stages[-1].setThresholds([1 - threshold_probabi
                                                threshold_probability
test_df_predictions = cv_model.transform(test_df)
metrics = evaluate_model(test_df_predictions, "TOTAL_FLOOR_AREA")
print(f"Metrics: {metrics}")
```

Результат:

```
[Stage 1984:================================================>
Metrics: {'precision': 0.9164163632014035, 'recall': 0.60013142262
```

2.3 Выводы

Обученная модель обладает не очень хорошим качеством. Для дальнейшего улучшения качества предсказания можно подобрать другую модель, сформировать другие признаки, добавить к модели дополнительные данные

ЗАКЛЮЧЕНИЕ

В заключении коротко приводятся и анализируются полученные результаты, предлагаются дальнейшие направления развития темы.

ПРИЛОЖЕНИЕ А

Пример листинга программного кода

Здесь можно привести полный листинг кода программы или модуля.

```
import matplotlib.pyplot as plt
import numpy as np

# Данные для графика
x = np.linspace(0, 10, 100)
y = np.sin(x)

# Создание графика
plt.figure(figsize=(10, 6))

plt.plot(x, y, label='sin(x)', color='blue', linewidth=2)

# Настройка графика
plt.title('График функции sin(x)', fontsize=16)
plt.xlabel('x', fontsize=14)
plt.ylabel('sin(x)', fontsize=14)
plt.legend()
plt.grid(True)

# Вывод графика
plt.show()
```

ПРИЛОЖЕНИЕ Б

Пример второго приложения

При необходимости, приложений может быть несколько.